Software Engineering Institute

Computer Forensics: Results of Live Response Inquiry vs. Memory Image Analysis

Cal Waits
Joseph Ayo Akinyele
Richard Nolan
Larry Rogers

August 2008

TECHNICAL NOTE
CMU/SEI-2008-TN-017

CERT Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

For information about SEI reports, please visit the publications section of our website (http://www.sei.cmu.edu/publications).

Table of Contents

List of Figures

Abstract

People responsible for computer security incident response and digital forensic examination need to continually update their skills, tools, and knowledge to keep pace with changing technology. No longer able to simply unplug a computer and evaluate it later, examiners must know how to capture an image of the running memory and perform volatile memory analysis using various tools, such as PsList, ListDLLs, Handle, Netstat, FPort, Userdump, Strings, and PSLoggedOn. This paper presents a live response scenario and compares various approaches and tools used to capture and analyze evidence from computer memory.

Introduction

It is no longer sufficient when gathering digital evidence to pull the plug and take the machine back to the lab. As technology continues to change, incident responders and digital forensic examiners must adopt new methods and tools to keep up. This is applicable especially in situations such as a live response scenario. For instance, with standard RAM size between two and eight gigabytes, the migration of malware into memory, and the increasing use of encryption by adversaries, it is no longer possible to ignore computer memory during an acquisition and subsequent analysis.

Traditionally, the only useful approach to investigating memory was a live response. This involved querying the system using API-style tools familiar to most network administrators. The first responder was looking for rogue connections or mysterious running processes. It was also possible to capture an image of the running memory, but until recently, short of a string search, it was difficult to gather useful data from a memory dump. The past few years have seen rapid development in tools focused exclusively on memory analysis.

This paper is organized into five sections:

Section 1 presents a scenario in which useful evidence can be collected from a running machine.

Section 2 describes a live response approach to the scenario.

Section 3 describes a volatile memory analysis approach to the scenario.

Section 4 discusses the drawbacks of both approaches and discusses which analysis approach provides a more viable investigation process.

Section 5 presents a conclusion.

1 Scenario

Sys-Internal vs. Memory Analysis Tools	Network Connections	Open Ports and Sockets	Running Processes	Hidden Running Processes	Terminated Processes	Loaded DLLs	Open Files	OS Kernel Modules	Process Dumps	Strings	User Logged On
Live Response											
PsList			X								
ListDLLs						X					
Handle							X				
Netstat	X	X									
Fport		X									
Userdump									X		
Strings										X	
PsLoggedOn											X
Memory Analysis											
Volatility	X	X	X	X	X	X	X	X			X
PTFinder			X	X	X						

Figure 1: Live response with Sys-Internal tools vs. memory analysis on a static memory dump

The traceability matrix of Figure 1 is a mapping of the capabilities of live response and memory analysis tools during an investigation of a memory image (or running memory). The Live Response part of Figure 1 lists the tools used in live response, and the Memory Analysis part shows tools that analyze physical memory dumps. This section contains hints for creating and maintaining Word files and suggestions for avoiding common mistakes.

In our virtual environment scenario, we start with a Windows XP Service Pack 2 virtual machine with an IP address of 192.168.203.132. Netcat was used to establish a telnet connection on port 4444 (PID: 3572) with a second machine at 192.168.203.133. MACSpoof was also installed and running (PID: 3008). This machine was then compromised by installing the FUTo rootkit and a ProRat server listening on port 5110. The netcat and MACSpoof processes were then hidden using the FUTo rootkit.

In the following sections, we present two possible techniques to approach the compromised system and we discuss what details are visible and invisible concerning the various compromises using each approach. The first approach we present is a live response process using sys-internal style tools. The second is a static memory dump analysis using open source memory analysis tools. Finally, we discuss the benefits and drawbacks of both approaches.

2 Live Response

The first approach is live response. Here an investigator would first establish a trusted command shell. In addition, they would establish a method for transmitting and storing the information on a data collection system of some sort. One option is to redirect the output of the commands on the compromised system to the data collection system. One popular tool is netcat, a network utility that transmits data across network connections. Another approach would be to insert a USB drive and write all query results to that external drive. Finally, investigators would attempt to bolster the credibility of the tool output in court. During a live interrogation of a system, it is important to realize that the state of the running machine is not static. This could lead to the same query producing different results based on when it is run. Therefore, hashing the memory is not effective. Rather, an investigator could compute a cryptographic checksum of the tool outputs and make a note of this hash value in the log. This would help dispel any notion that the results had been altered after the fact. In this exercise, HELIX (a live response and Linux bootable CD), was used to establish a trusted command shell.

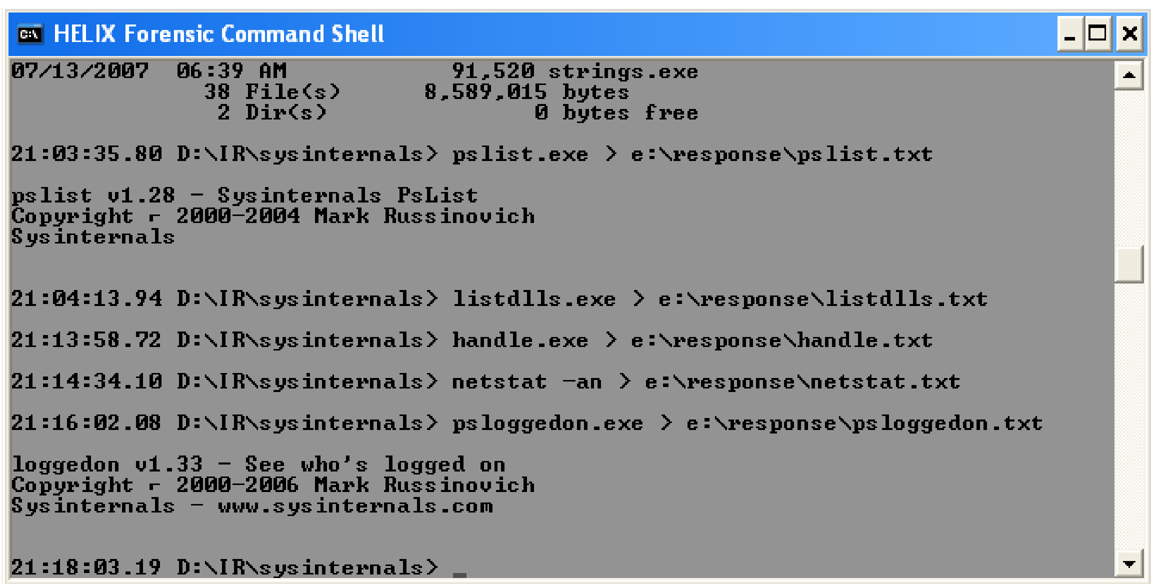

Figure 2: Trusted command shell established using HELIX

Once the above data collection setup is complete, an investigator can begin to collect evidence from the compromised system. The sys-internal style tools used in this exercise are not meant to be an exhaustive list. Rather, they are representative of the types of tools available. The common thread for the tools used is that each relies on native API calls to some degree, and thus the results are filtered through the operating system. The tools used in this case were PsList, ListDLLs, Handle, Netstat, FPort, Userdump, Strings, and PSLoggedOn.

Name	Pid	Pri	Thd	Hnd	Priv	CPU Time	Elapsed Time
Idle	0	0	1	0	0	1:45:29.406	0:00:00.000
System	4	8	56	495	0	0:00:52.765	0:00:00.000
smss	608	11	3	21	168	0:00:00.234	22:28:01.462
csrss	656	13	12	490	2044	0:00:23.718	22:28:00.274
winlogon	680	13	19	564	7788	0:00:03.984	22:27:59.274
services	724	9	16	364	3996	0:02:06.984	22:27:57.883
lsass	736	9	19	344	3720	0:00:02.093	22:27:57.712
vmacthlp	896	8	1	24	704	0:00:00.093	22:27:56.446
svchost	912	8	17	194	3060	0:00:00.515	22:27:56.133
svchost	1016	8	11	283	1804	0:00:01.125	22:27:53.618
svchost	1112	8	71	1352	14516	0:00:19.328	22:27:52.899
svchost	1168	8	6	81	1292	0:00:01.109	22:27:52.508
svchost	1308	8	15	215	1748	0:00:00.296	22:27:52.258
ccSetMgr	1508	8	6	188	4040	0:00:00.593	22:27:50.915
ccEvtMgr	1552	8	15	286	4220	0:00:00.609	22:27:50.196
SPBBCSvc	1640	8	14	239	6188	0:00:01.281	22:27:49.571
spoolsv	1716	8	11	116	3528	0:00:00.359	22:27:49.274
Rtvscan	648	8	51	579	59672	0:00:50.687	22:27:42.305
VMwareService	1224	13	3	56	1004	0:00:06.468	22:27:40.415
explorer	2468	8	11	425	16392	0:01:18.656	22:26:39.665
VMwareTray	2616	8	1	27	764	0:00:00.250	22:26:34.602
VMwareUser	2632	8	3	154	2212	0:00:04.375	22:26:34.477
ccApp	2640	8	9	240	4260	0:00:00.531	22:26:34.430
wuauclt	3072	8	3	164	2188	0:00:00.359	22:26:23.821
cmd	3540	8	1	31	2036	0:00:00.796	22:25:19.290
cmd	3796	8	1	31	2024	0:00:00.281	1:03:51.489
services	3144	8	3	85	15068	0:01:54.062	0:59:25.329
cmd	3816	8	1	31	1996	0:00:00.109	0:14:52.184
helix	3872	8	9	289	21496	0:00:10.796	0:06:40.437
cmd	1896	8	1	31	2020	0:00:00.171	0:01:12.078
pslist	3656	13	2	82	1188	0:00:00.375	0:00:03.781

Figure 3: Results from pslist

PsList allows investigators to view process and thread statistics on a system. Applying PsList reveals all running processes on the system but does not reveal the presence of the rootkit or the other processes that the rootkit has hidden (netcat and MACSpoof).

```
ListDLLs v2.25 - DLL lister for Win9x/NT
Copyright (C) 1997-2004 Mark Russinovich
Sysinternals - www.sysinternals.com

------------------------------------------------------------------------------
System pid: 4
Command line: <no command line>
------------------------------------------------------------------------------
smss.exe pid: 608
Command line: \SystemRoot\System32\smss.exe

  Base        Size      Version         Path
  0x48580000  0xf000                    \SystemRoot\System32\smss.exe
  0x7c900000  0xb0000   5.01.2600.2180  C:\WINDOWS\system32\ntdll.dll
------------------------------------------------------------------------------
csrss.exe pid: 656
Command line: C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows
SharedSection=1024,3072,512 Windows=On SubSystemType=Windows ServerDll=basesrv,1
ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2
ProfileControl=Off MaxRequestThreads=16

  Base        Size      Version         Path
  0x4a680000  0x5000                    \??\C:\WINDOWS\system32\csrss.exe
  0x7c900000  0xb0000   5.01.2600.2180  C:\WINDOWS\system32\ntdll.dll
  0x75b40000  0xb000    5.01.2600.2180  C:\WINDOWS\system32\CSRSRV.dll
  0x75b50000  0x10000   5.01.2600.2180  C:\WINDOWS\system32\basesrv.dll
  0x75b60000  0x4b000   5.01.2600.3103  C:\WINDOWS\system32\winsrv.dll
  0x77f10000  0x47000   5.01.2600.3316  C:\WINDOWS\system32\GDI32.dll
```

Figure 4: Excerpt from ListDLLs output

ListDLLs allows investigators to view the currently loaded DLLs for a process. Applying ListDLLs reveals the DLLs loaded by all running processes. However, since there are processes that are hidden, ListDLLs cannot show the DLLs loaded for them. Thus, critical evidence that could reveal the presence of the rootkit is missed. The problem is that an attacker may have compromised the Windows API upon which an investigator’s toolkit depends. To a degree, this is the case with our scenario. As a result, rootkit manipulation cannot be easily detected with these tools. A more sophisticated and non-intrusive approach is necessary to find what could be critical evidence.

```
  284: File  (RW-)   C:\WINDOWS\WindowsUpdate.log
------------------------------------------------------------------------------
cmd.exe pid: 3540 USER-D6520207A3\Administrator
   64: File  (RW-)   C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b
   84: Section       \BaseNamedObjects\ShimSharedMemory
   88: File  (RW-)   C:\tools\nc111nt
------------------------------------------------------------------------------
cmd.exe pid: 3796 USER-D6520207A3\Administrator
   64: File  (RW-)   C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b
   78: File  (RW-)   C:\tools\FUTo_enhanced\FUTo_enhanced\FUTo\EXE
   84: Section       \BaseNamedObjects\ShimSharedMemory
```

Figure 5: Excerpt from Handle output

The Handle utility allows investigators to view open handles for any process. It reveals the open files for all the running processes, which includes the path to the file. In this case, one of the command shells is running from a directory labeled …\FUTo\EXE. This is a strong hint of the presence of the FUTo rootkit. Similarly, there is another instance of cmd.exe running from C:\tools\nc11nt. The nc11nt folder is a default for the windows distribution of netcat. While it is useful to show the implications of the tool results, it is important to remember that simply renaming these directories or running the cmd.exe from a different directory would have prevented these disclosures.

```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5112           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5757           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:51100          0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1033         0.0.0.0:0              LISTENING
  TCP    192.168.203.132:139    0.0.0.0:0              LISTENING
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:1026           *:*
  UDP    0.0.0.0:1054           *:*
  UDP    0.0.0.0:4500           *:*
  UDP    127.0.0.1:123          *:*
  UDP    127.0.0.1:1900         *:*
  UDP    192.168.203.132:123    *:*
  UDP    192.168.203.132:137    *:*
  UDP    192.168.203.132:138    *:*
  UDP    192.168.203.132:1900   *:*
```

Figure 6: Netstat results

The Netstat utility allows investigators to view the network connections of a running machine. Nestat (with the –an option) reveals nothing immediately suspicious in this case.

```
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
1016                 ->  135   TCP
4     System         ->  139   TCP
4     System         ->  445   TCP
2640  ccApp          ->  1033  TCP   C:\Program Files\Common Files\Symantec Shared\ccApp.exe
0     System         ->  1130  TCP
3144  services       ->  5112  TCP   C:\WINDOWS\services.exe
3144  services       ->  5757  TCP   C:\WINDOWS\services.exe
3144  services       ->  51100 TCP   C:\WINDOWS\services.exe

0     System         ->  123   UDP
2640  ccApp          ->  123   UDP   C:\Program Files\Common Files\Symantec Shared\ccApp.exe
0     System         ->  137   UDP
0     System         ->  138   UDP
1016                 ->  445   UDP
4     System         ->  500   UDP
3144  services       ->  1026  UDP   C:\WINDOWS\services.exe
3144  services       ->  1054  UDP   C:\WINDOWS\services.exe
0     System         ->  1900  UDP
3144  services       ->  4500  UDP   C:\WINDOWS\services.exe
```

Figure 7: Results of FPort

FPort allows investigators to view all open TCP/IP and UDP ports and maps them to each process, which includes the PID and the executable path. In our scenario, FPort does not reveal the presence of the connections hidden by the rootkit.

Userdump allows investigators to extract the memory dumps of running processes for offline analysis. Since it has a specific meta-data format, dumpchk.exe (http://support.microsoft.com/kb/315271) is normally used to verify that a usable process memory dump was produced. The Strings utility extracts ASCII and UNICODE characters from binary files. In this case, an investigator would apply it to the process dumps and see what evidence can be uncovered.

Finally, PsLoggedOn helps investigators discover users who have logged in both locally and remotely. In this case, only the Administrator is logged on.

3 Volatile Memory Analysis

The second approach is volatile memory image analysis. It is similar to live response, in that an investigator would first establish a trusted command shell. Then they would establish a data collection system and a method for transmitting the data. However, an investigator would only acquire a physical memory dump of the compromised system and transmit it to the data collection system for analysis. In this case VMware allows investigators to simply suspend the virtual machine and use the .vmem file as a memory image. As established in digital forensic practices, an investigator would also compute the hash upon completion of the memory capture. Unlike traditional hard drive forensics, no hash is calculated for memory before acquisition. Due to the volatile nature of running memory, the imaging process is taking a snapshot of a "moving target."

The primary difference between this approach and Live Response is that no additional evidence is needed on the compromised system. Therefore, the evidence can be analyzed on the collection system.

As seen in Figure 1, we discuss the capabilities of two memory analysis tools applied on the memory image. We also describe what evidence is visible to an investigator in this type of analysis. The tools used are The Volatility Framework by Volatile Systems and PTFinder by Andreas Schuster. The capabilities of each tool are discussed as well as the information it extracts from memory dumps. These tools are recent additions to the excellent array of open source resources available to digital investigators. There are other memory analysis tools not included in this comparison.

3.1 VOLATILITY

The Volatility Framework is a collection of command-line python script that analyzes Windows XP Service Pack 2 memory images. It allows an investigator to interrogate the image in a style similar to that used during a live response. Volatility is distributed under a GNU General Public License. For this exercise version 1.1.2 was used. It allows an investigator to interrogate the image in a style similar to that used during a live response. Commands available in the 1.1.2 version include `ident`, `datetime`, `pslist`, `psscan`, `thrdscan`, `dlllist`, `modules`, `sockets`, `sockscan`, `connections`, `connscan`, `vadinfo`, `vaddump`, and `vadwalk`. Several of the commands used during the exercise are explained below.

Using the syntax `python volatility ident -f WinXP_victim.vmem` and `python volatility datetime -f WinXP_victim.vmem,` the `ident` and `datetime` commands are used to gather information about the image itself (in this case the image used was the WinXP_victim.vmem file). The first provides the operating system type, virtual address translation mechanism, and a starting directory table base (DTB), while the second reports the date and time the image was captured. This provides valuable information because it assists with documentation purposes in a digital investigation. Furthermore, it is useful for creating a timeline of events with other pieces of evidence in the digital investigation.

Name	Pid	PPid	Thds	Hnds	Time
System	4	0	55	405	Thu Jan 01 00:00:00 1970
smss.exe	608	4	3	21	Wed Jul 09 21:36:12 2008
csrss.exe	656	608	12	473	Wed Jul 09 21:36:13 2008
winlogon.exe	680	608	19	515	Wed Jul 09 21:36:14 2008
services.exe	724	680	16	363	Wed Jul 09 21:36:15 2008
lsass.exe	736	680	18	343	Wed Jul 09 21:36:16 2008
vmacthlp.exe	896	724	1	24	Wed Jul 09 21:36:17 2008
svchost.exe	912	724	16	192	Wed Jul 09 21:36:17 2008
svchost.exe	1016	724	9	270	Wed Jul 09 21:36:20 2008
svchost.exe	1112	724	75	1323	Wed Jul 09 21:36:20 2008
svchost.exe	1168	724	6	81	Wed Jul 09 21:36:21 2008
svchost.exe	1308	724	15	212	Wed Jul 09 21:36:21 2008
ccSetMgr.exe	1508	724	7	190	Wed Jul 09 21:36:22 2008
ccEvtMgr.exe	1552	724	16	288	Wed Jul 09 21:36:23 2008
SPBBCSvc.exe	1640	724	15	243	Wed Jul 09 21:36:24 2008
spoolsv.exe	1716	724	11	116	Wed Jul 09 21:36:24 2008
Rtvscan.exe	648	724	52	581	Wed Jul 09 21:36:31 2008
VMwareService.e	1224	724	3	56	Wed Jul 09 21:36:33 2008
explorer.exe	2468	2392	13	504	Wed Jul 09 21:37:34 2008
VMwareTray.exe	2616	2468	1	27	Wed Jul 09 21:37:39 2008
VMwareUser.exe	2632	2468	4	156	Wed Jul 09 21:37:39 2008
ccApp.exe	2640	2468	10	242	Wed Jul 09 21:37:39 2008
wuauclt.exe	3072	1112	4	166	Wed Jul 09 21:37:49 2008
cmd.exe	3540	2468	1	31	Wed Jul 09 21:38:54 2008
cmd.exe	3796	2468	1	31	Thu Jul 10 19:00:22 2008
services.exe	3144	3132	3	85	Thu Jul 10 19:04:48 2008
cmd.exe	3816	2468	1	31	Thu Jul 10 19:49:21 2008
cmd.exe	2032	1224	0	-1	Thu Jul 10 19:52:21 2008

Figure 8: Results from Volatility pslist command

Using the `pslist` command produces results similar to SysInternal pslist.exe tool used during the live response.

```
System pid: 4
Unable to read PEB for task.
************************************************************************
smss.exe pid: 608
Command line : \SystemRoot\System32\smss.exe

Base         Size         Path
0x48580000   0xf000       \SystemRoot\System32\smss.exe
0x7c900000   0xb0000      C:\WINDOWS\system32\ntdll.dll

************************************************************************
csrss.exe pid: 656
Command line : C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows SharedSe
ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllI

Base         Size         Path
0x4a680000   0x5000       \??\C:\WINDOWS\system32\csrss.exe
0x7c900000   0xb0000      C:\WINDOWS\system32\ntdll.dll
0x75b40000   0xb000       C:\WINDOWS\system32\CSRSRV.dll
0x75b50000   0x10000      C:\WINDOWS\system32\basesrv.dll
0x75b60000   0x4b000      C:\WINDOWS\system32\winsrv.dll
0x77f10000   0x47000      C:\WINDOWS\system32\GDI32.dll
```

Figure 9: Volatility dlllist results

This is also the case with the `dlllist` command. This option shows the size and path to all the DLLs used by each running process.

```
Fast
No.  PID    Time created             Time exited              Offset     PDB        Remarks
---- ------ ------------------------ ------------------------ ---------- ---------- ----------------

   1      0                                                   0x00551d80 0x0031a000 Idle
   2   3816 Thu Jul 10 19:49:21 2008                          0x01f4e020 0x1a5402a0 cmd.exe
   3   3540 Wed Jul 09 21:38:54 2008                          0x01fb25e0 0x1a540400 cmd.exe
   4   3072 Wed Jul 09 21:37:49 2008                          0x01fdb5b0 0x1a540320 wuauclt.exe
   5   3144 Thu Jul 10 19:04:48 2008                          0x01fef020 0x1a540460 services.exe
   6   2616 Wed Jul 09 21:37:39 2008                          0x02056020 0x1a540100 VMwareTray.exe
   7    648 Wed Jul 09 21:36:31 2008                          0x020fcda0 0x1a540240 Rtvscan.exe
   8    724 Wed Jul 09 21:36:15 2008                          0x0217b5a8 0x1a540080 services.exe
   9   1716 Wed Jul 09 21:36:24 2008                          0x02190da0 0x1a540200 spoolsv.exe
  10   1168 Wed Jul 09 21:36:21 2008                          0x021a6990 0x1a540160 svchost.exe
  11   1508 Wed Jul 09 21:36:22 2008                          0x021a9020 0x1a5401a0 ccSetMgr.exe
  12   2032 Thu Jul 10 19:52:21 2008 Thu Jul 10 19:52:22 2008 0x021d0358 0x1a540220 cmd.exe
  13   2632 Wed Jul 09 21:37:39 2008                          0x021d93a0 0x1a540360 VMwareUser.exe
  14   2640 Wed Jul 09 21:37:39 2008                          0x021fdda0 0x1a540380 ccApp.exe
  15    736 Wed Jul 09 21:36:16 2008                          0x0235a020 0x1a5400a0 lsass.exe
  16   2468 Wed Jul 09 21:37:34 2008                          0x0235e7b0 0x1a540340 explorer.exe
  17   1552 Wed Jul 09 21:36:23 2008                          0x023edda0 0x1a5401c0 ccEvtMgr.exe
  18   1640 Wed Jul 09 21:36:24 2008                          0x023fd9a0 0x1a5401e0 SPBBCSvc.exe
  19    656 Wed Jul 09 21:36:13 2008                          0x02401c08 0x1a540040 csrss.exe
  20    680 Wed Jul 09 21:36:14 2008                          0x024205f0 0x1a540060 winlogon.exe
  21   1308 Wed Jul 09 21:36:21 2008                          0x0242c888 0x1a540180 svchost.exe
  22   1112 Wed Jul 09 21:36:20 2008                          0x0242cb08 0x1a540140 svchost.exe
  23   1016 Wed Jul 09 21:36:20 2008                          0x02436b10 0x1a540120 svchost.exe
  24    912 Wed Jul 09 21:36:17 2008                          0x0243e250 0x1a5400e0 svchost.exe
  25   1224 Wed Jul 09 21:36:33 2008                          0x02458da0 0x1a540260 VMwareService.e
  26   3796 Thu Jul 10 19:00:22 2008                          0x02490170 0x1a5402e0 cmd.exe
  27    896 Wed Jul 09 21:36:17 2008                          0x024a86a8 0x1a5400c0 vmacthlp.exe
  28      0 Thu Jul 10 18:59:55 2008                          0x024fb020 0x1a5402c0 MACSpoof.exe
  29    608 Wed Jul 09 21:36:12 2008                          0x02578408 0x1a540020 smss.exe
  30      4                                                   0x025c8830 0x0031a000 System
```

Figure 10: *Volatility psscan command shows MACSpoof.exe*

However, when we use the `psscan` option something new is revealed. With a PID of 0 MACSpoof.exe shows up in the list. This command scans for, and returns, the physical address space for the all EPROCESS objects found.

```
Local Address             Remote Address            Pid
------------------------- ------------------------- ------

127.0.0.1:1114            127.0.0.1:1033            3144
192.168.203.132:1076      64.236.22.201:80          1004
192.168.203.132:4444      192.168.203.133:2867      3572
127.0.0.1:1033            127.0.0.1:1114            2640
```

Figure 11: *Volatility connscan resutls*

While netstat failed to provide any sign of the netcat activity, using `connscan` shows us the connection with 192.168.203.133 on port 4444. The results also indicate a PID of 3572 associated with this connection. The fact that this PID is missing from the other queries could indicate the presence of a rootkit.

```
Name                                                        Base
\WINDOWS\system32\ntkrnlpa.exe                              0x804d7000
\WINDOWS\system32\hal.dll                                   0x806ce000
\WINDOWS\system32\KDCOM.DLL                                 0xf8b9a000
\WINDOWS\system32\BOOTVID.dll                               0xf8aaa000
ACPI.sys                                                    0xf856b000
\WINDOWS\system32\DRIVERS\WMILIB.SYS                        0xf8b9c000
.
.
.
\SystemRoot\system32\DRIVERS\USBSTOR.SYS                    0xf8a9a000
\SystemRoot\system32\drivers\kmixer.sys                     0xf57aa000
\??\C:\tools\FUTo_enhanced\FUTo_enhanced\FUTo\EXE\msdirectx.sys   0xf5b05000
```

Figure 12: Excerpt of results from Volatility modules option

The Volatility Framework also allows an investigator to list all the kernel modules loaded at the time the memory image was captured. While the path of the last entry from the `modules` command certainly attracts attention, an even less obvious path would show the msdirectx.sys. A simple Google search will show this module is associated with rootkits.

The Volatility Framework provided evidence about the attacker's IP address and the connections to the system. In addition, it provided some leads in terms of the possibility of a rootkit and hidden process being present.

NOTE:

After this article was written a new version, Volatility 1.3, was released. Volatility now supports Windows XP SP2 and SP3 as well as Linux operating systems. Several new modules have also been added, increasing the capabilities of the framework significantly.

3.2 PTFINDER

The second memory analysis tool, PTFinder, is a Perl script that supports analysis of Windows 2000/2003/XP/XP SP2 operating system versions. PTFinder enumerates processes and threads in a memory dump. PTFinder uses a brute force approach to enumerating the processes and uses various rules to determine whether the information is either a legitimate process or just bytes. Although this tool does not reveal anything new in terms of malware, it does reflect a benefit of volatile memory analysis, which is repeatability of the results.

No.	Type	PID	TID	Time created	Time exited	Offset	CR3	Remarks
1	Proc	0				0x00551d80	0x0031a000	Idle
2	Proc	3816		2008-07-10 19:49:21		0x01f4e020	0x1a5402a0	cmd.exe
3	Proc	3540		2008-07-09 21:38:54		0x01fb25e0	0x1a540400	cmd.exe
4	Proc	3072		2008-07-09 21:37:49		0x01fdb5b0	0x1a540320	wuauclt.exe
5	Proc	3144		2008-07-10 19:04:48		0x01fef020	0x1a540460	services.exe
6	Proc	2616		2008-07-09 21:37:39		0x02056020	0x1a540100	VMwareTray.exe
7	Proc	648		2008-07-09 21:36:31		0x020fcda0	0x1a540240	Rtvscan.exe
8	Proc	724		2008-07-09 21:36:15		0x0217b5a8	0x1a540080	services.exe
9	Proc	1716		2008-07-09 21:36:24		0x02190da0	0x1a540200	spoolsv.exe
10	Proc	1168		2008-07-09 21:36:21		0x021a6990	0x1a540160	svchost.exe
11	Proc	1508		2008-07-09 21:36:22		0x021a9020	0x1a5401a0	ccSetMgr.exe
12	Proc	2032		2008-07-10 19:52:21	2008-07-10 19:52:22	0x021d0358	0x1a540220	cmd.exe
13	Proc	2632		2008-07-09 21:37:39		0x021d93a0	0x1a540360	VMwareUser.exe
14	Proc	2640		2008-07-09 21:37:39		0x021fdda0	0x1a540380	ccApp.exe
15	Proc	736		2008-07-09 21:36:16		0x0235a020	0x1a5400a0	lsass.exe
16	Proc	2468		2008-07-09 21:37:34		0x0235e7b0	0x1a540340	explorer.exe
17	Proc	1552		2008-07-09 21:36:23		0x023edda0	0x1a5401c0	ccEvtMgr.exe
18	Proc	1640		2008-07-09 21:36:24		0x023fd9a0	0x1a5401e0	SPBBCSvc.exe
19	Proc	656		2008-07-09 21:36:13		0x02401c08	0x1a540040	csrss.exe
20	Proc	680		2008-07-09 21:36:14		0x024205f0	0x1a540060	winlogon.exe
21	Proc	1308		2008-07-09 21:36:21		0x0242c888	0x1a540180	svchost.exe
22	Proc	1112		2008-07-09 21:36:20		0x0242cb08	0x1a540140	svchost.exe
23	Proc	1016		2008-07-09 21:36:20		0x02436b10	0x1a540120	svchost.exe
24	Proc	912		2008-07-09 21:36:17		0x0243e250	0x1a5400e0	svchost.exe
25	Proc	1224		2008-07-09 21:36:33		0x02458da0	0x1a540260	VMwareService.e
26	Proc	3796		2008-07-10 19:00:22		0x02490170	0x1a5402e0	cmd.exe
27	Proc	896		2008-07-09 21:36:17		0x024a86a8	0x1a5400c0	vmacthlp.exe
28	Proc	0		2008-07-10 18:59:55		0x024fb020	0x1a5402c0	MACSpoof.exe
29	Proc	608		2008-07-09 21:36:12		0x02578408	0x1a540020	smss.exe
30	Proc	4				0x025c8830	0x0031a000	System

Figure 13: Results of PTFinder with the "no threads" option

The "no threads" option on PTFinder gives us a list of processes found in the memory dump. Notice the MACSpoof.exe near the bottom of the list with a PID of 0.

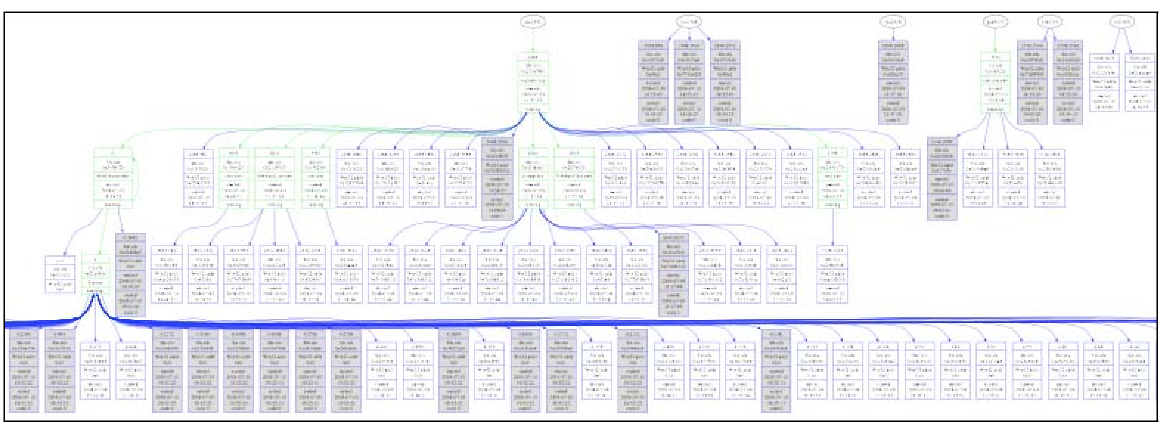

Figure 14: PTFinder graph of threads and processes

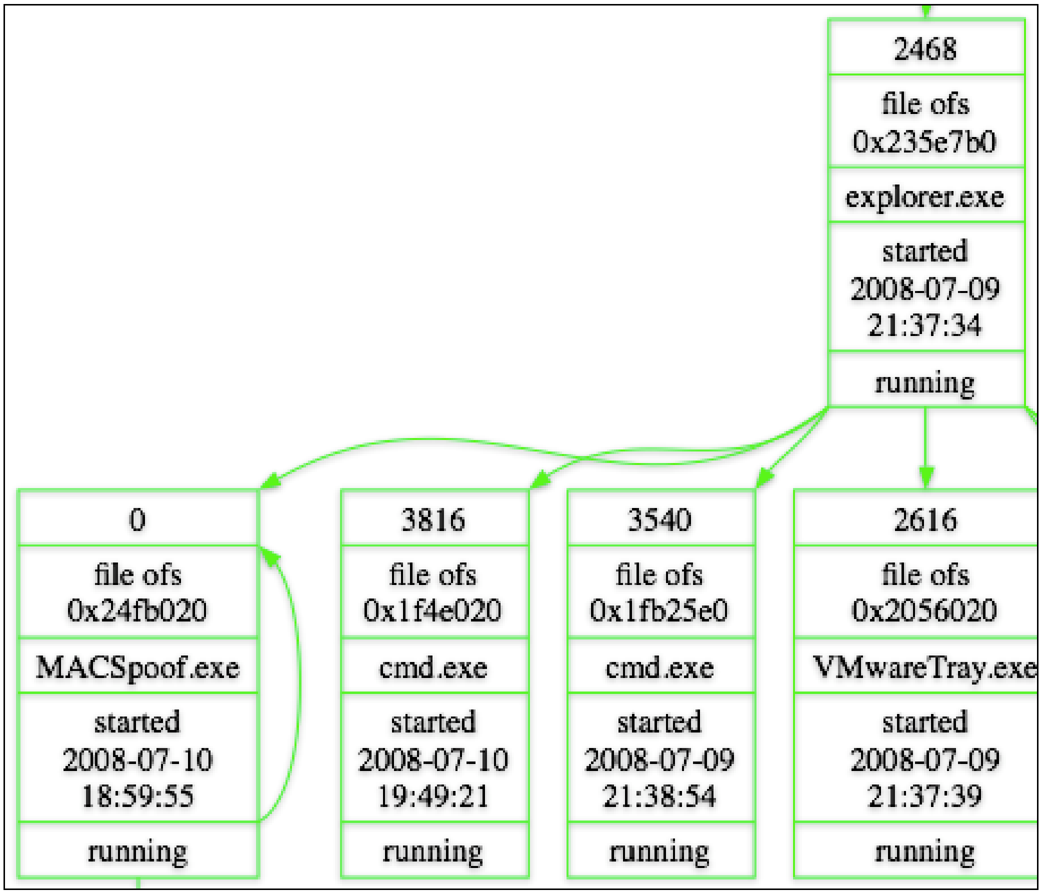

Figure 15: Close up of PTFinder graph showing the MACSpoof.exe

PTFinder also has the ability to output results in the dot(1) format. This is an open source graphics language that provides a visual representation of the relationships between threads and processes. (These relationships are shown in full in Figure 14 and close up in Figure 15.)

4 Analysis

Thus far, we have described two different incident response approaches to the scenario discussed in Section 1.2. The first approach is the well-known live response where an investigator surveys the crime scene, collects the evidence, and at the same time probes for suspicious activity. The second approach is the relatively new field of volatile memory analysis where an investigator collects the memory dump and performs analysis in an isolated environment. In both approaches, we described what types of information gave an investigator insight into the scenario. Now, we will discuss some of the issues with live response that hinder effective analysis of a digital crime scene. We will also discuss why volatile memory analysis should be the ideal approach to investigating cyber crime.

While the purpose of live response is to collect all relevant evidence from the system that will likely be used to confirm whether an incident occurred, the implementation of the process has significant setbacks, including the following:

- First Responder toolkit may rely on Windows API: The problem is that if an attacker compromises the system and changes system files without an investigator suspecting, then an investigator could collect a large amount of evidence that is based on compromised sources. As a result, this would damage the credibility of the analysis in a court of law.
- Live response is not repeatable: The information in memory is volatile and with every passing second, bytes are being overwritten. As we saw in our scenario, the tools may produce the correct output and in themselves can be verified by a third-party expert. However, the input data supplied to them can never be reproduced. As a result, this puts the evidence collected at risk in a court of law. Therefore, it becomes difficult for investigators to prove the correctness of their analysis of the evidence. [Walters 2007].
- Investigators cannot ask new questions later: The live response process does not support examination of the evidence in a new way. This is mainly because the same inputs to the tools from the collection phase cannot be reproduced. As a result, investigators cannot ask new questions later on in the analysis phase of the investigation [Walters 2007]. By the analysis phase, it becomes impossible to learn anything new about the compromise. In addition, as we saw in our scenario, once critical evidence is missed during collection, it can never be recovered again. It damages the case against the attacker.

On the other hand, a volatile memory analysis shows promise in that the only source of evidence is the physical memory dump. Moreover, collection of physical memory has become more commonly practiced. An investigator can then build the case by analyzing the memory dump in an isolated environment that is non-obtrusive to the evidence. Thus, volatile memory analysis addresses the drawbacks facing live response as follows:

- It limits impact to the compromised system: Unlike live response, memory analysis uses a simplified approach to investigating a crime scene. It involves merely extracting the memory dump and minimizes the fingerprint left on the compromised system. In addition, the nature of live response puts the analysis of the evidence at risk in a court of law. As a result, an investigator gets the added benefit of analyzing the memory dump fully confident that the impact to the data is minimal.
- Analysis is repeatable: Since the memory dumps are analyzed directly and in isolated environments, this allows for multiple sources to validate and repeat the analysis. We saw this in our scenario, where the hidden malware processes were identified by the two tools. In addition, it allows for conclusions made by investigators to be verified by third-party experts. Essentially, it improves the credibility of the analysis in a court of law.
- Nature of analysis supports asking new questions later: Contrary to live response, memory analysis allows investigators with more expertise, technique, or understanding to ask new questions later on in the investigation [Walters 2007]. We saw this in our scenario. Our initial analysis of the memory dump with Volatility gave us some suspicion of a rootkit being present on the system. We later confirmed this with evidence of the terminated rootkit process using the Lsproc script. This important evidence may have been missed in a live response.

One of the greatest drawbacks with volatile memory analysis is that the tools' support has not matured enough. This is because with every release of a new operating system, the physical memory structure changes. Development of memory analysis tools has been gaining velocity recently, but the kinks still remain. This is an emerging field and new ground is being broken across the area of study.

5 Conclusions

Despite the drawbacks associated with volatile memory analysis, it is the authors' opinion that volatile memory analysis will be integral to the digital investigation process going forward. Based on current technologies, the best approach is a hybrid based on situational awareness and a triage mentality. This same triage mentality is aptly demonstrated by emergency medical personnel when dealing with multiple casualties. For instance, the EMT must make a rapid assessment of accident victims before deciding on the priority and type of treatment. Instead of being used to gather exhaustive amounts of data, live response should move to a triage role, collecting just enough information to determine the next appropriate step. Full memory analysis (and the requisite memory acquisition) should be used to augment and supplement traditional digital forensic examination when greater understanding of the running state of the machine is critical to resolving the case. In other words, no response scenario will be identical, so it is impractical to build rigid procedures and checklists for use in the field.

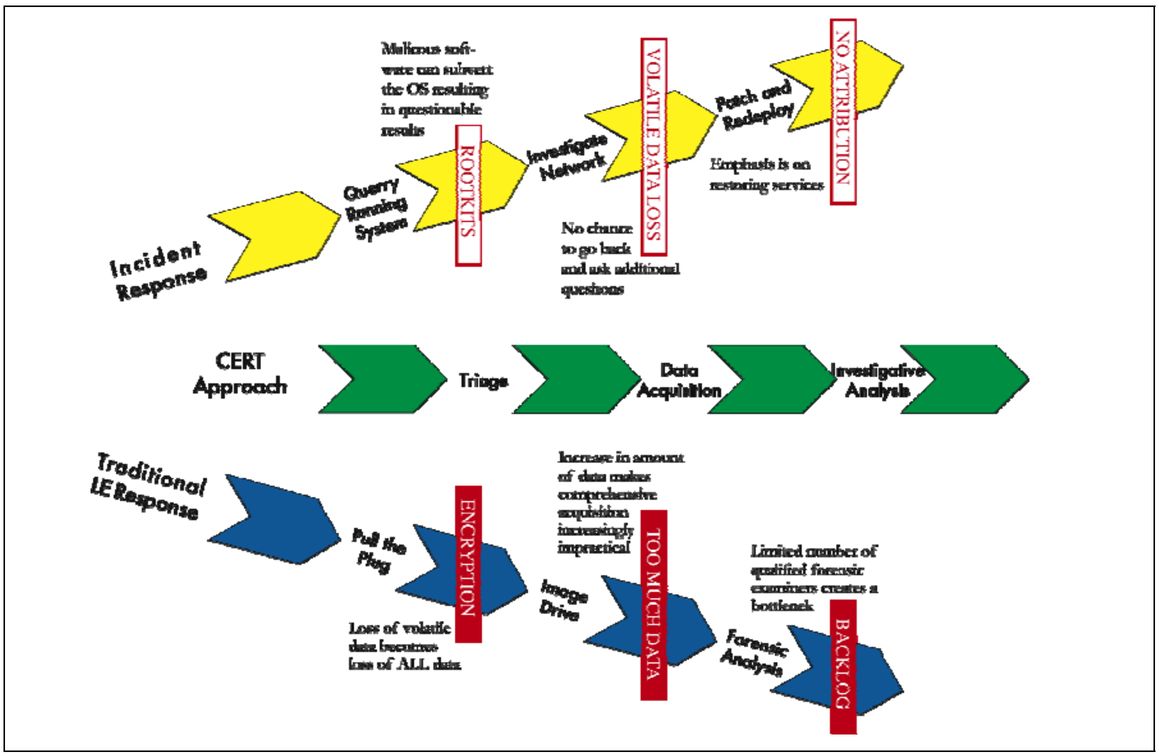

Figure 16: Hybrid approach

Rather, an informed policy should be developed that takes into consideration the various types of scenarios that may necessitate a live response, memory acquisition, or simple power-off. This approach falls somewhere between a typical law enforcement response and the techniques used by IT staff during incident response.

Figure 2 illustrates the nature of the “middle ground” approach. In one example, live system investigation may be necessary to determine the presence of mounted encrypted containers or full-disk encryption. If detected, the examiner would then switch to capturing a memory image for off-line analysis (as well as capturing the data in an unencrypted state). A memory image allows for the application of analysis tools, now or later, which can extract valuable cryptographic material.

References

URLs are valid as of the publication date of this document.

[SysInternals]
http://technet.microsoft.com/en-us/sysinternals/default.aspx

[Volatility]
https://www.volatilesystems.com/default/volatility

[PTFinder]
http://computer.forensikblog.de/en/2007/11/ptfinder_0_3_05.html

[Walters 2007]
Walters A. & Petroni N. *Volatools: Integrating Volatile Memory Forensics into the Digital Investigation Process Black Hat*. DC 2007. February 2007.

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE August 2008		3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Computer Forensics: Results of Live Response Inquiry vs. Memory Image Analysis			5. FUNDING NUMBERS FA8721-05-C-0003
6. AUTHOR(S) Cal Waits, Joseph Ayo Akinyele, Richard Nolan, Larry Rogers			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2008-TN-017
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116			10. SPONSORING/MONITORING AGENCY REPORT NUMBER Error! No text of specified style in document.
11. SUPPLEMENTARY NOTES			
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS			12B DISTRIBUTION CODE
13. ABSTRACT (MAXIMUM 200 WORDS) People responsible for computer security incident response and digital forensic examination need to continually update their skills, tools, and knowledge to keep pace with changing technology. No longer able to simply unplug a computer and evaluate it later, examiners must know how to capture an image of the running memory and perform volatile memory analysis using various tools, such as PsList, ListDLLs, Handle, Netstat, FPort, Userdump, Strings, and PSLoggedOn. This paper presents a live response scenario and compares various approaches and tools used to capture and analyze evidence from computer memory.			
14. SUBJECT TERMS computer security, incident response, computer forensics, volatile memory analysis, live response			15. NUMBER OF PAGES 31
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102